PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30

A2

(11) International Publication Number: WO 00/58871

(43) International Publication Date: 5 October 2000 (05.10.00)

(21) International Application Number: PCT/US00/08346

(22) International Filing Date: 30 March 2000 (30.03.00)

(30) Priority Data:
09/282,806 31 March 1999 (31.03.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 09/282,806 (CIP)
Filed on 31 March 1999 (31.03.99)

(71) Applicant *(for all designated States except US)*: AMERICA ONLINE, INC. [US/US]; 22000 AOL Way, Dulles, VA 20166 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: HENDREN, C., Hudson, III [US/US]; 1340 Old Grade Road, Strasburg, VA 22657 (US).

(74) Agents: HAYDEN, John, F. et al.; Fish & Richardson, P.C., 601 Thirteenth Street N.W., Washington, DC 20005 (US).

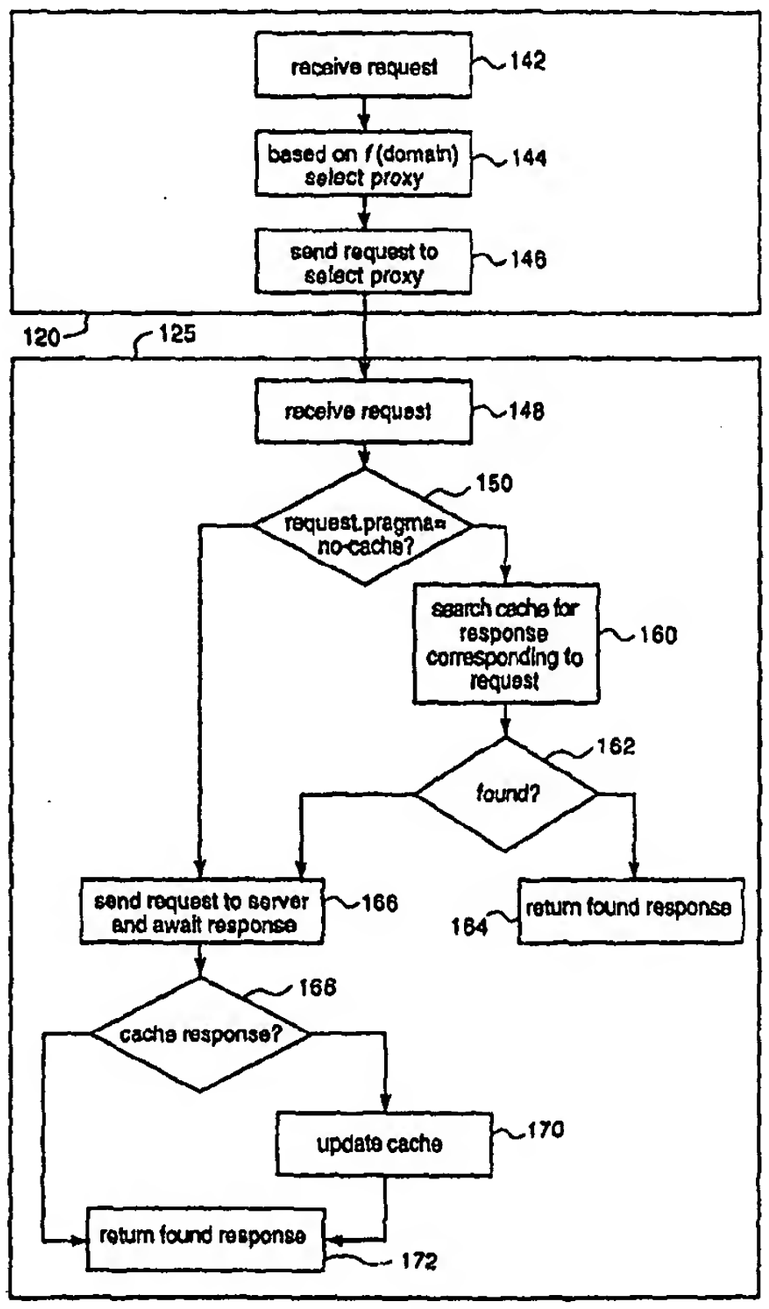
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SELECTING A CACHE

(57) Abstract

Methods, systems, and computer program products for selecting one of a plurality of caches that store information received from network sites. The inventions feature receiving information that identifies the location of a resource within a domain and selecting a cache based on the information that identifies the location of the resource within the domain.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SELECTING A CACHE

Technical Field

This invention relates to selecting a cache that stores information received from a network site.

Background

Computer networks such as the Internet provide users with a powerful tool for acquiring and distributing information. Since the emergence of the World Wide Web in the early 1990s, users have flocked to the Internet in growing numbers. The corresponding increase in network traffic, however, has increased the length of time users must wait to receive information. During busy periods, users commonly wait several minutes for complex Web-pages to load.

Many computers on the World Wide Web communicate using HTTP (HyperText Transfer Protocol). HTTP defines a client/server relationship between clients requesting resources (e.g., HTML (HyperText Markup Language) documents, audio, video, graphics, executable or interpreted instructions, and other information) and servers offering those resources. As shown in FIG. 1, a client 100 transmits a request for a resource 104 to a server 102 providing the resource 104. The server then transmits a response that can include the requested resource 104 along with other information such as any errors that may have occurred. Software running on the client 100 (e.g., a browser) can present the retrieved resource 104 to the user.

As shown in FIG. 2, an HTTP request 106 includes a URI (Universal Resource Indicator) 108 (e.g., a URL (Universal Resource Locator)) that identifies a requested resource 104 within a hierarchical location scheme. That is, the URI 108 describes a resource with increasing specificity, for example, first by identifying the domain 116 (e.g., "www.domain.com") providing the requested resource 104, then by identifying one or more directories 117 (e.g., "/directory/subdirectory") within the domain 116, and finally by identifying a file 118 (e.g., "filename.html") within the identified set of directories 117.

The HTTP request 106 also can include other information such as the type of client 110 making the request (e.g., a Microsoft® Internet Explorer browser), the preferred

language of a user 112, and other information 114. A request 106 can vary in size from a few bytes to several kilobytes.

The exchange shown in FIG. 1 is a simplification of network communication. In fact, a request typically passes through many intermediate agents before reaching a server 102. One type of intermediate agent is a proxy 120. As shown in FIG. 3, a proxy 120 receives requests from a client 100 and optionally sends them on to the server 102 providing a requested resource. The proxy 120 receives the server's response 108 and can send the response 108 on to the client 100. The proxy 120 can perform many functions in addition to acting as a conduit for client 100 / server 102 communication. For example, by examining information stored in requests and/or responses, the proxy 120 can act as a filter, for example, by intercepting mature content before it reaches a client 100 used by a child.

As shown in FIG. 4, many different users often request the same resource (e.g., Web-page). Thus, storing commonly requested resources in a cache 126 can reduce the amount of time it takes to provide a response to a request. As shown, a cache database table 128 stores client requests 130 and previously received server responses 132 to these requests 130. The table 128 also can store an expiration date 134 for a stored response 132 and other information 136. Different cache functions handle storage and retrieval of information from the cache.

As shown in FIG. 5, a proxy 120 (e.g., a proxy at an ISP (Internet Service Provider)) initially receiving a request can forward the request to a cache proxy 124 that includes a cache database 126 and instructions that implement cache functions 125. These can functions 125 search, read, and write the cache database 126. When the cache proxy 124 receives a request, the cache proxy 124 searches the cache database 126 for a corresponding response.

Referring to FIG. 6, if a response corresponding to the request previously has been stored in the cache 124, the cache proxy 124 can return the response without accessing the server 102 from which the requested resource originally came. Eliminating transmission of the request from the proxy 120 to the server 102 and the corresponding

transmission of a response from the server 102 back to the proxy 120 reduces client 100 access times and network traffic.

As shown in FIG. 7, if the cache 126 does not store a previous response to a request, the cache proxy 124 transmits a request to the server 102. Alternatively, the cache proxy 124 can transmit a request to the server 102 if the request includes a "pragma = no-cache" directive indicating that the response provided should not be retrieved from a cache. Regardless of whether a cache search failed or a request included a "pragma = no-cache" directive, the cache proxy 124 may store the response provided by the server 106 for future use.

As shown in FIG. 8, a proxy 120 may access multiple cache proxies 124, 138, 140, for example, cache proxies collected within the same ISP 122. This capability enables a single proxy 120 to access a very large number of cached responses. The proxy 120 routes a request received from a client to one of the cache proxies 124, 138, 140 by hashing (e.g., transforming information into a number) the domain 116 included in the URI 108 of the request. For example, hashing a domain of "www.a.com" may yield a "1" while hashing a domain of "www.b.com" may yield a "2." These requests can be sent to cache proxy 124 and 138, respectively. This scheme collects the different resources provided by the same domain into the same cache proxy. For example, "www.a.com/a.html" and "www.a.com/b.html" will share the same domain and reside on the same cache 124

As described above, a cache proxy 124, 138, 140 may not previously have cached a response corresponding to a particular request. In such a case, the cache proxy 124 transmits the request to the server providing a particular resource. For example, as shown, a request for "www.c.com/c" is routed to cache proxy #2 140 based on the request's URI domain information ("www.c.com"). The cache proxy 140, however, must transmit the request to the server 102 providing the resource since the cache does not yet store "www.c.com/c." Upon receipt of the response, the cache proxy 140 can store "www.c.com/c" in its cache for future use.

To summarize, as shown in FIG. 9, a proxy 120 using multiple cache proxies receives a request 142 and performs 144 a hash function on the domain information included in the URI of the request. Based on the hash results, the proxy 120 transmits 146 the request to one of the cache proxies 124, 138, and 140.

The cache proxy 124, 138, 140 receiving 148 the request can determine whether to search its cache 150. If the cache proxy searches 160 and finds 162 a response corresponding to the request in its cache, the cache proxy 124, 138, 140 can return 164 the found response to the proxy 120. If the cache proxy decided 150 not to search its cache or failed 162 in its search for the request, the cache proxy sends 166 the request on to the server identified by the request URI. After the cache proxy receives the response, the cache proxy can determine 168 whether to store 170 the response in its cache to speed future requests. The cache proxy then returns 172 the received response to the proxy 120 for transmission to the client making the request.

The present inventors recognized that the method of distributing responses among caches described above can result in a distribution that underutilizes the caches.

Summary

In general, in one aspect, a method of selecting one of a plurality of caches that store information received from at least one network site includes receiving information that identifies the location of a resource within a domain and selecting a cache based on the information that identifies the location of the resource within the domain.

Embodiments may include one or more of the following features. Receiving information may include receiving a request such as an HTTP (HyperText Transfer Protocol) request. The information may include a URI (Universal Resource Indicator) (e.g., a URL (Universal Resource Locator)) identifying the location of a resource. Selecting a cache may be based on the domain of the resource in addition to the location of a resource within the domain. Selecting a cache may include use of a hashing function. Selecting a cache may include selecting a cache proxy. The method may also include sending a request to the selected cache proxy.

The information identifying the location of a resource within a domain can include one or more directories and/or a file name.

In general, in another aspect, a method of selecting one of a plurality of caches that store information received from a network site includes receiving information that identifies a location of a resource expressed using a hierarchical location scheme that includes identifiers corresponding to different hierarchical levels, and selecting a cache based on information identifiers that correspond to more than one hierarchical level.

Embodiments may include one or more of the following features. A hierarchical level may be a domain. A hierarchical level may be the location of a resource within a domain.

In general, in another aspect, a method of selecting one of a plurality of caches that store information received from a network site includes receiving information that identifies a location of a resource; and selecting a cache based on all the received information identifying the location of the resource.

In general, in another aspect, a method of selecting one of a plurality of caches that store information received from at least one network site includes receiving an HTTP (HyperText Transfer Protocol) request that includes a URI (Universal Resource Indicator) identifies the location of a resource within a domain and selecting a cache proxy by hashing the URI domain and URI information that identifies the location of the resource within the domain. The method further includes sending a request to the selected cache proxy.

In general, in another aspect, a computer program product, disposed on a computer readable medium, for selecting one of a plurality of caches that store information received from at least one network site, the computer program product includes instructions for causing a processor to: receive information that identifies the location of a resource within a domain, and select a cache based on the information that identifies the location of the resource within the domain.

In general, in another aspect, a system for handling requests for information provided by a network server includes a plurality of cache proxies and a front-end proxy. The front-end proxy includes instructions for causing the front-end proxy processor to receive information that identifies the location of a resource within a domain, and select a cache based on the information that identifies the location of the resource within the domain.

Advantages may include one or more of the following. Performing a hash that includes the resource information of a URI spreads storage of resources provided by a particular domain across multiple caches. Because a handful of domains receive the lion's share of requests (e.g., "www.aol.com"), spreading the resources provided by these domains over multiple caches enables more efficient use of the caches as each cache reads

and writes a substantially equal number of requests and responses. Thus, no one cache becomes overloaded with request processing while other caches remain underutilized.

Modifying the instructions of a proxy instead of modifying the instructions executed by cache proxies reduces the difficulty of incorporating these techniques into an existing network configuration.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

Drawing Descriptions

FIG. 1 is a flow diagram of a client/server request/response exchange.

FIG. 2 is a diagram of an HTTP request.

FIG. 3 is a flow diagram of requests and responses exchanged between a client, server, and proxy.

FIG. 4 is a diagram of a cache for storing server responses.

FIG. 5 is a diagram of a proxy and cache proxy.

FIG. 6 is a flow diagram illustrating retrieval of a response from a cache proxy.

FIG. 7 is a flow diagram illustrating storage of a response in a cache proxy.

FIG. 8 is a flow diagram illustrating operation of multiple cache proxies.

FIG. 9 is a flow chart of a process for using multiple cache proxies.

FIG. 10 is a diagram of a proxy that includes instructions for determining whether to bypass cache functions based on a request.

FIG. 11 is a flow diagram of a proxy bypassing cache functions.

FIG. 12 is a flow diagram of a proxy that selects a cache based on information that indicates the location of a resource within a domain.

FIG. 13 is a flow chart of proxy instructions for bypassing cache functions and selecting a cache.

FIG. 14 is a flow chart of proxy instructions for determining whether to bypass cache functions.

Like reference numbers and designations in the various drawings indicate like elements.

Detailed Description

Referring to FIG. 10, a proxy 174 uses multiple caches 126 to store server responses. As shown, the caches 126 and cache functions 125 are included in cache proxies 124, 138, 140, 141. Because communications between the proxy 174 and a cache proxy 124, 138, 140, 141 can conform to HTTP standards, encapsulating a cache 124 and its associated functions 125 in a cache proxy offers programmers a simple method of integrating cache functions into existing processing. A cache proxy, however, is not necessarily required to implement cache functions (e.g., instructions that search, read, or write cache information). For example, a single monolithic proxy could manage different caches without using independent cache proxies.

Although caches can reduce the amount of time required to produce a response for a given request, the use of a cache is not without costs. For example, searching the cache for a particular request can be time consuming. When such a search is unsuccessful, the use of a cache actually increases the amount of time taken to process a request. In many cases, the extra time spent searching a cache unsuccessfully and storing a server's subsequent response is justified by the time savings enjoyed by future requests. However, as will be discussed, this is not always the case.

As shown in FIG. 10, the proxy 174 includes instructions 176 that determine whether to bypass cache functions 125 based on a received request. Conditional use of cache functions 125 enables the proxy 174 to identify situations where cache functions 125 are likely to slow processing of a current request without a compensating reduction in the time needed to process future requests. For example, attributes of a request may indicate that a cache is unlikely to have previously stored a corresponding response. The request's attributes may further indicate that any response provided by a server is unlikely to be added to the cache for future use. In such circumstances, executing cache functions 125 offers neither short-term nor long-term reductions in access times. As a result of this cost-benefit analysis, the conditional use of cache functions enables a proxy to provide the benefits of cache functions without needlessly suffering their potential inefficiencies.

Referring to FIG. 11, upon receiving a request, the instructions 176 for the proxy 174 determine whether to bypass caching functions based on the request. If the proxy 174 determines not to use cache functions 125, the proxy 174 sends a request to the

server 104. Bypassing the caches 124, 138, 140, 141 saves the amount of time needed to search a cache and to store a response in the cache database.

Bypassing the cache proxies 124, 138, 140, 141 also reduces the number of agents that process a request. This eliminates the computational overhead that would otherwise be added by a cache proxy that processes a request. For example, each agent typically parses a received request to extract information. While parsing is conceptually straightforward, parsing can be computationally intensive if a request is large and/or includes variable length data. Hence, eliminating parsing performed by a cache proxy 124, 138, 140, 141 can produce surprisingly large reductions in access times.

Referring to FIG. 12, if the instructions 176 for proxy 174 decide to use cache functions, the proxy 174 sends the request to a cache proxy 124, 138, 140, 141. The proxy 174 selects a cache based at least in part on information 117, 118 (FIG. 2) included in the URI of a request that identifies the location of a resource within a domain. For example, the proxy 174 could select a cache based on the resource location (e.g., "directory/subdirectory/a.html") or the resource location in addition to the URI domain (e.g., "www.domain.com/directory/subdirectory/a.html"). For example, the proxy 174 can implement a hash function that transforms a complete URI into a number. For example, a hash function could add the ASCII (American Standard Code for Information Interchange) values of all the characters in the URI and modulo divide by the number of caches. Hashing to a number between 1 and a prime number is believed to produce an even distribution of values. Hence, a system that includes a prime number of caches may evenly distribute responses among the caches. Other cache functions can easily be used instead of the sample function described above. The hash described above and others can be used in systems that do not use a prime number of caches.

Based on the results of the hash function, the proxy instructions can select a cache. For example, if the hash of "www.c.com/a" yielded "0" and an Internet Service Provider had four cache proxies, the proxy 174 could send the request to cache proxy #0 124.

Selecting a cache based on information that indicates the location of a resource within a domain distributes different resources provided by a server 104 across multiple caches 124, 138, 140, 141. For example, the resources "a" and "b" provided by "www.a.com" are cached by cache proxies 140 and 138, respectively. Because a handful

of domains receive the large majority of requests (e.g., "www.NewYorkTimes.com"), spreading the resources provided by these servers over multiple caches enables more efficient use of the caches as each cache reads and writes a substantially equal number of requests and responses. Thus, the hashing scheme enforces load balancing and no cache becomes overloaded with requests while other caches are underutilized.

Referring also to FIG. 13, after a proxy receives a request 142, the proxy instructions 176 determine 178 whether or not to bypass cache functions 125. If the proxy instructions 176 decide to bypass cache functions 125, the proxy sends 184 the request on to the server and awaits the server's response for subsequent transmission back to the client.

If the proxy instructions 176 decide not to bypass cache functions 125, the proxy instructions 176 select 180 a cache based on resource information included in the URI of the request. The proxy instructions 176 send the request to the selected cache 182. As the bypassing and selection instructions are implemented by the proxy 174, the cache proxies 124, 138, 140, 141 need not be altered. As a result, the cache bypassing and selection mechanisms are transparent to the cache proxies. This enables an administrator to integrate the bypassing and cache selection techniques in an existing network of proxies with a minimum of reconfiguration.

Referring to FIG. 14, the proxy 174 can determine whether to bypass caching functions by examining attributes of a received request. For example, an HTTP request can identify itself as being a "POST" or "GET" request. A "POST" request that "posts" (i.e., sends) data to a server such as information a user entered into a web-page form. A "GET" request "gets" a resource identified by a URI (e.g., "GET www.domain.com/a.html"). A "GET" request can include parameters for processing by a server. Such "GET" requests include information following a "?" delimiter. For example, "GET www.a.com/cgi-bin/ProcessName.cgi?Name=JohnDoe" sends the "Name=JohnDoe" as a parameter.

The HTTP specification mandates that responses to both "POST" and "GET" requests that include parameters must not be cached. This prevents users from receiving inaccurate information. For example, if a first POST request includes data entered by a user into an order form for widgets, the server's response "thank you for your order of widgets" should not be cached. If such a response was cached, the user's second order of

widgets would not reach the server but nevertheless result in a cached response of "thank you for your order of widgets."

Thus, these requests can neither be satisfied by accessing a cache nor can responses to these requests add to the information stored in a cache. By making a single determination to bypass caching functions 125 based on whether a request is a POST request 186 or a GET request that includes parameter 188, a proxy 174 can reduce the amount time needed to service a request.

An administrator may choose to bypass cache functions based on other request information. For example, because CGI (Common Gateway Interface) script responses often expire very quickly and often produce different responses for the same requests, an administrator may elect to bypass cache functions for requests including the letters "cgi" in the request URI.

The methods and techniques described here may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus embodying these techniques may include appropriate input and output devices, a computer processor, and a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor. A process embodying these techniques may be performed by a programmable processor executing a program of instructions to perform desired functions by operating on input data and generating appropriate output. The techniques may advantageously be implemented in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to. a data storage system, at least one input device, and at least one output device. Each computer program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language may be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and

removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

A number of embodiments of the present invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, the distribution of the functions and components need not be as shown, but can instead be distributed over any number of computers or networks. Additionally, although we use the terms client and server, any given program may be capable of acting as either a client or server; our use of these terms may refer only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Accordingly, other embodiments are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of selecting one of a plurality of caches that store information received from at least one network site, the method comprising:

receiving information that identifies the location of a resource within a domain; and

selecting a cache based on the information that identifies the location of the resource within the domain.

2. The method of claim 1, wherein selecting a cache based on the information comprises selecting a cache at least in part on the information that identifies the location of the resource within the domain.

3. The method of claim 1, wherein receiving information comprises receiving a request.

4. The method of claim 3, wherein receiving a request comprises receiving an HTTP (HyperText Transfer Protocol) request.

5. The method of claim 1, wherein the information that identifies the location of a resource comprises information included in a URI (Universal Resource Indicator).

6. The method of claim 5, wherein the URI comprises a URL (Uniform Resource Locator).

7. The method of claim 1, further comprising selecting a cache based on the domain of the resource in addition to the location of a resource within the domain.

8. The method of claim 1, wherein selecting a cache comprises hashing.

9. The method of claim 8, wherein hashing comprises adding the ASCII (American Standard Code for Information Interchange) values of characters.

10. The method of claim 1, wherein selecting a cache comprises selecting a cache proxy.

11. The method of claim 10, further comprising sending a request to the selected cache proxy.

12. The method of claim 1, wherein the information identifying the location of a resource within a domain comprises at least one directory.

13. The method of claim 1, wherein the information identifying the location of a resource within a domain comprises a file name.

14. The method of claim 1, wherein the information identifying the location of a resource consists of one or more directories and a file name.

15. A method of selecting one of a plurality of caches that store information received from a network site, the method comprising:

receiving information that identifies a location of a resource, the information expressed using a hierarchical location scheme that includes identifiers corresponding to different hierarchical levels; and

selecting a cache based on information identifiers that correspond to more than one hierarchical level.

16. The method of claim 15, wherein a first hierarchical level comprises a domain and a second hierarchical level comprises the location of a resource within a domain.

17. A method of selecting one of a plurality of caches that store information received from a network site, the method comprising:

receiving information that includes a URI (Universal Resource Indicator) of a resource; and

selecting a cache based on the entire URI.

18. A method of selecting one of a plurality of caches that store information received from at least one network site, the method comprising:

receiving an HTTP (HyperText Transfer Protocol) request that includes a URI (Universal Resource Indicator) identifies the location of a resource within a domain;

selecting a cache proxy by hashing the URI domain and URI information that identifies the location of the resource within the domain; and

sending a request to the selected cache proxy.

19. A computer program product, disposed on a computer readable medium, for selecting one of a plurality of caches that store information received from at least one network site. the computer program product including instructions for causing a processor to:

receive information that identifies the location of a resource within a domain; and

select a cache based on the information that identifies the location of the resource within the domain.

20. The computer program product of claim 19, wherein the instructions that cause the processor to select a cache based on the information comprise instructions that cause the processor to select a cache at least in part on the information.

21. The computer program product of claim 19, wherein the instructions that cause the processor to receive information comprise instructions that cause the processor to receive a request.

22. The computer program product of claim 19, wherein the information that identifies the location of a resource comprises information included in a URI (Universal Resource Indicator).

23. The computer program product of claim 19, further comprising instructions for causing the processor to select a cache based on the domain of the resource in addition to the location of a resource within the domain.

24. The computer program product of claim 19, wherein the instructions for causing the processor to select a cache comprise instructions for causing the processor to hash .

25. The computer program product of claim 19, wherein the instructions for causing the processor to select a cache comprise instructions for causing the processor to select a cache proxy.

26. The computer program product of claim 19, further comprising instructions for causing the processor to send a request to the selected cache proxy.

27. A system for handling requests for information provided by a network server, the system comprising:

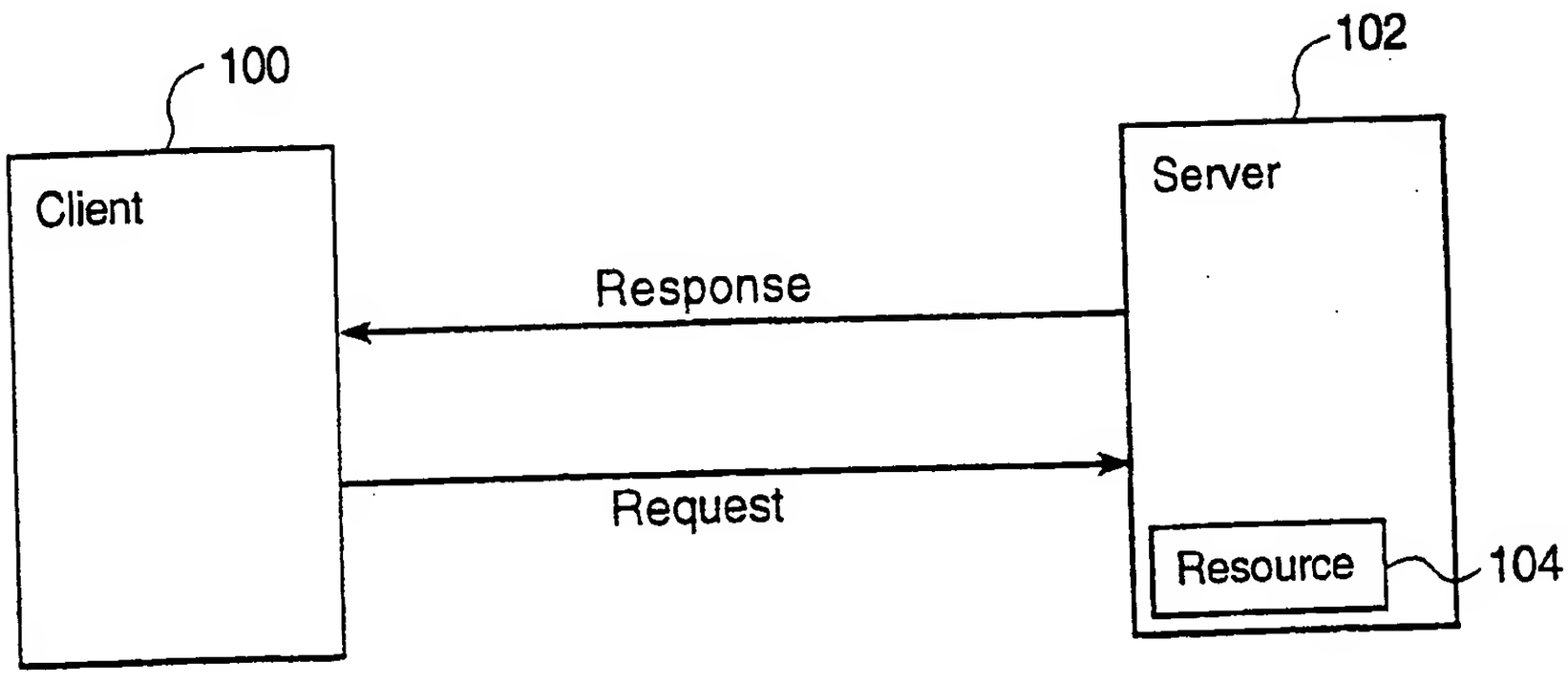
a plurality of cache proxies; and

a front-end proxy, the front-end proxy including instructions for causing the front-end proxy processor to:

receive information that identifies the location of a resource within a domain; and

select a cache based on the information that identifies the location of the resource within the domain.

FIG. 1



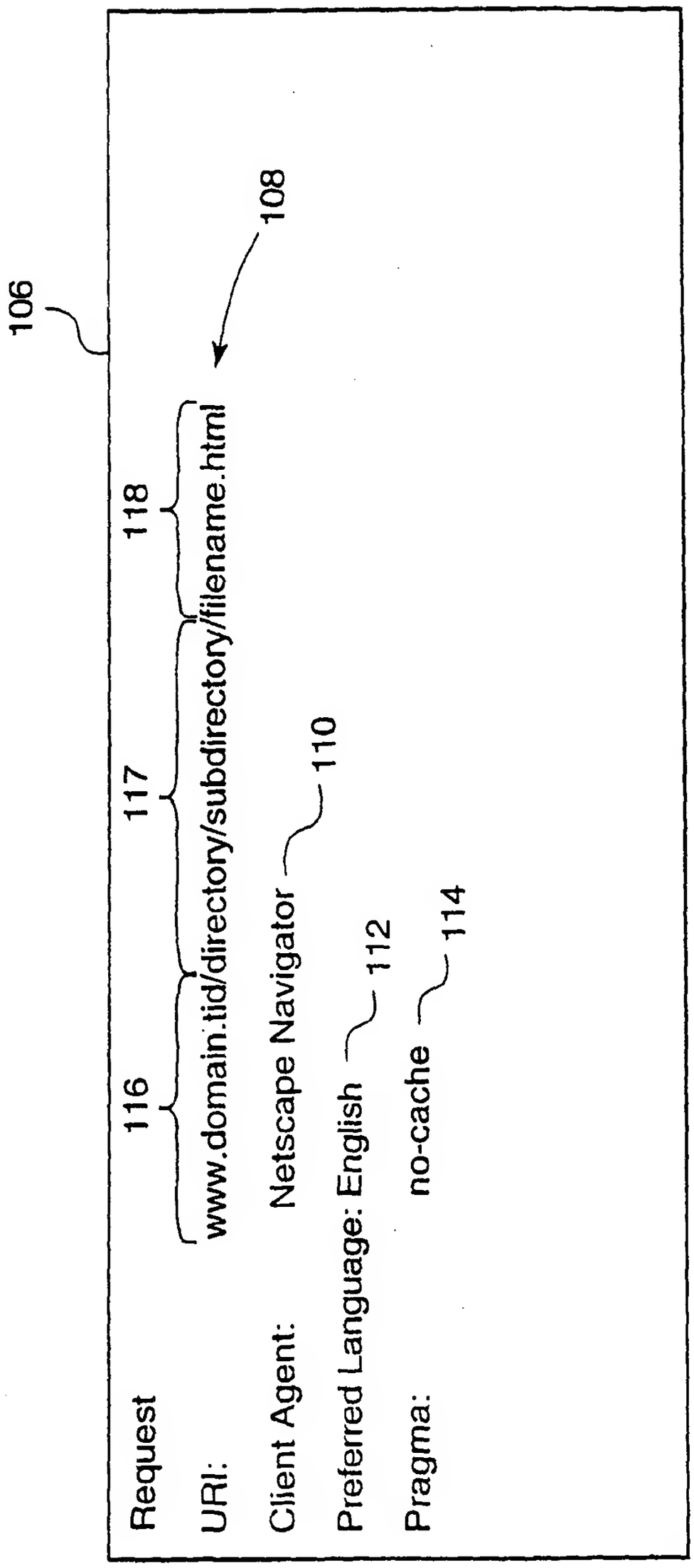

106
108
Request
URI:
www.domain.tid/directory/subdirectory/filename.html
117
118
Client Agent: Netscape Navigator
110
Preferred Language: English
112
Pragma: no-cache
114
116

FIG. 2

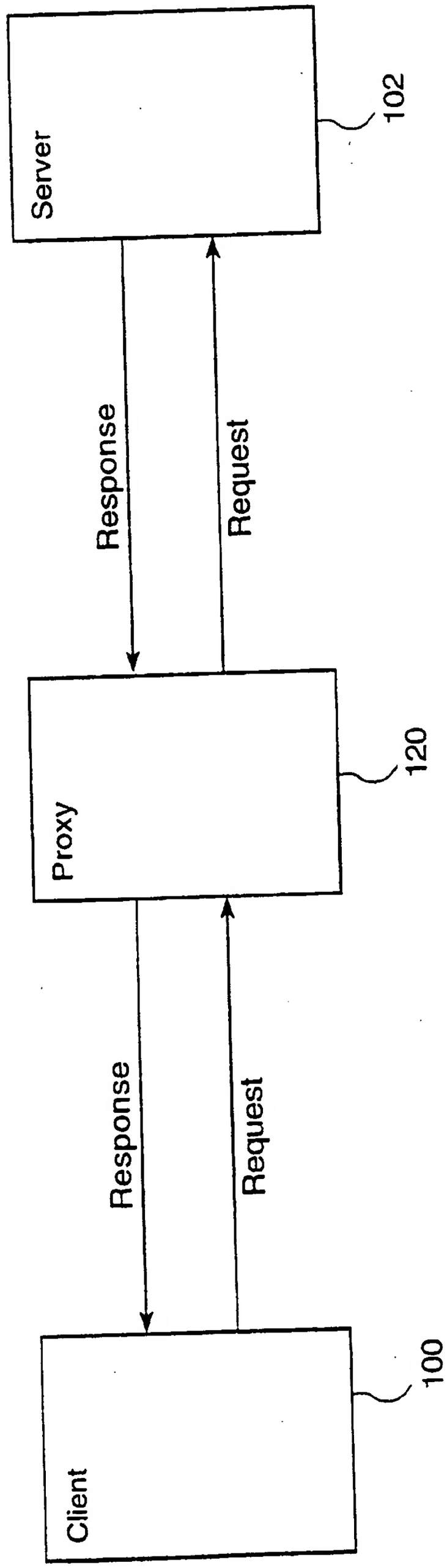

Server
102
Response
Request
Proxy
120
Response
Request
Client
100

FIG. 3

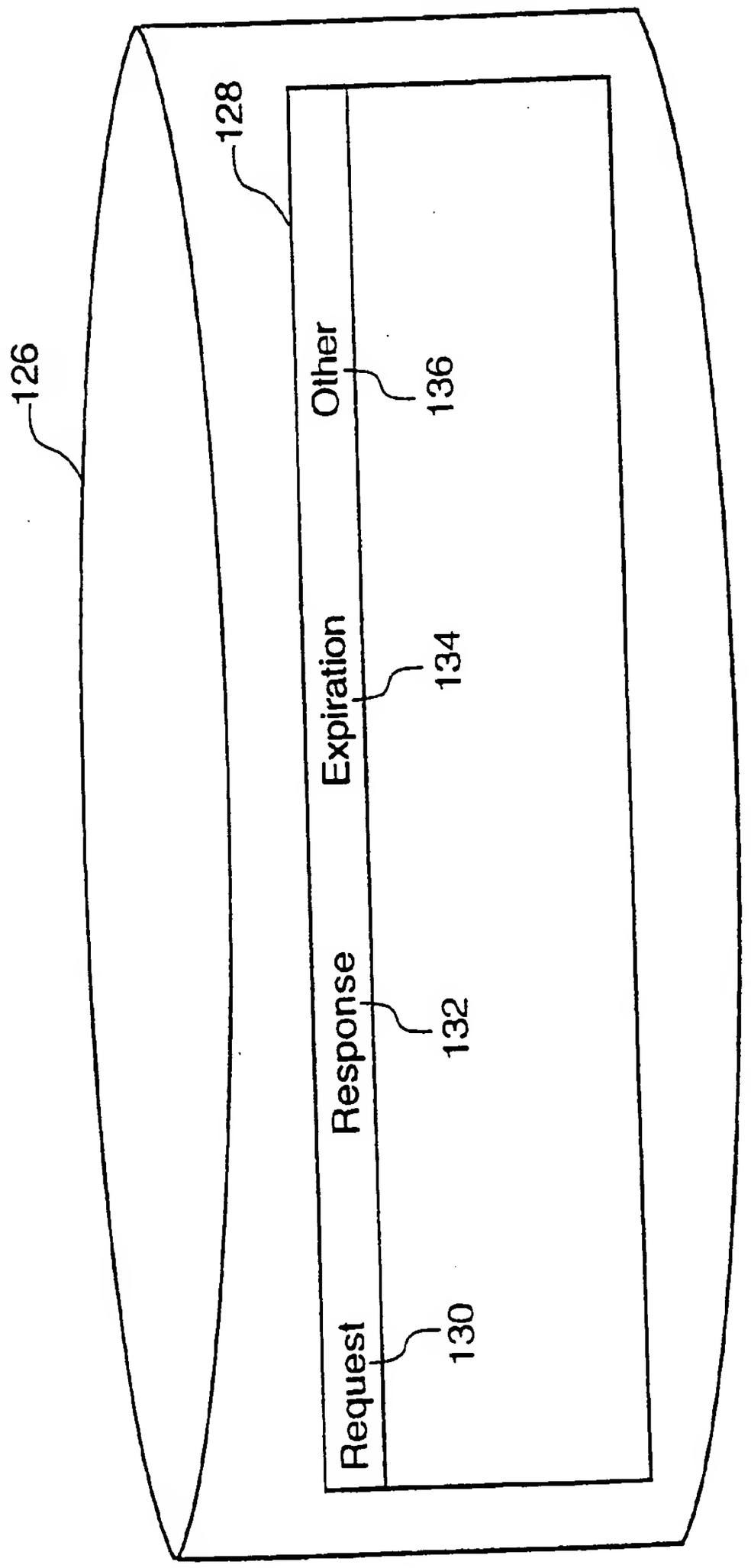


FIG. 4

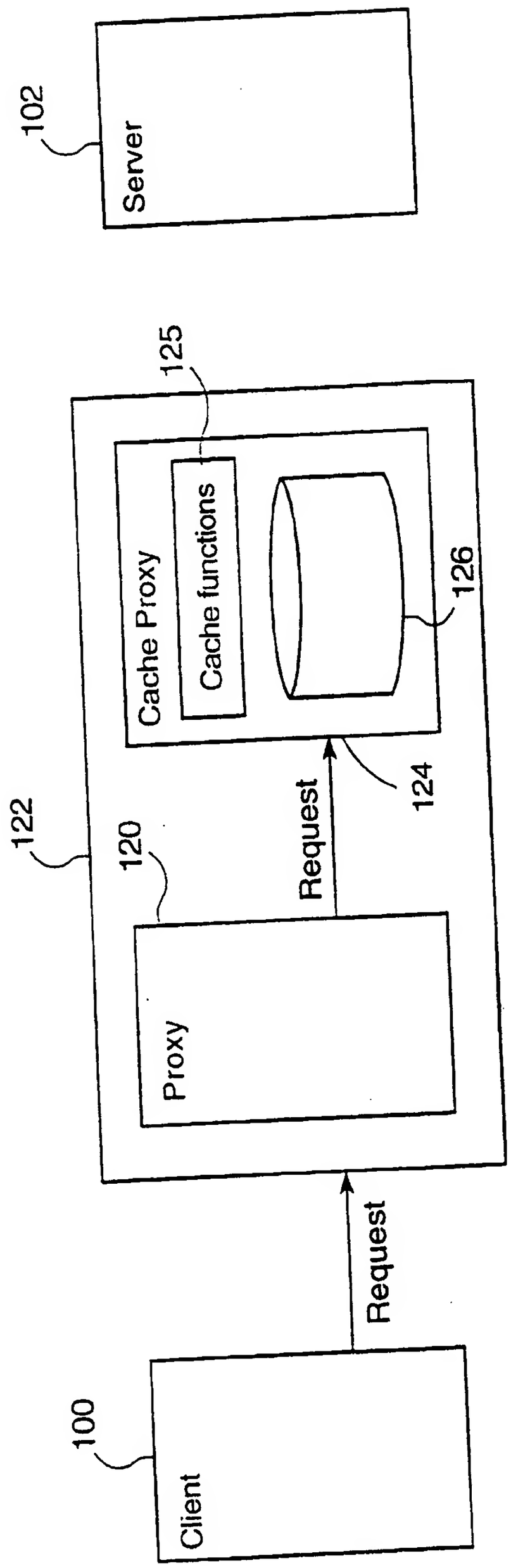

Server
102
Cache Proxy
Cache functions
126
124
Request
Proxy
120
122
Request
Client
100

FIG. 5

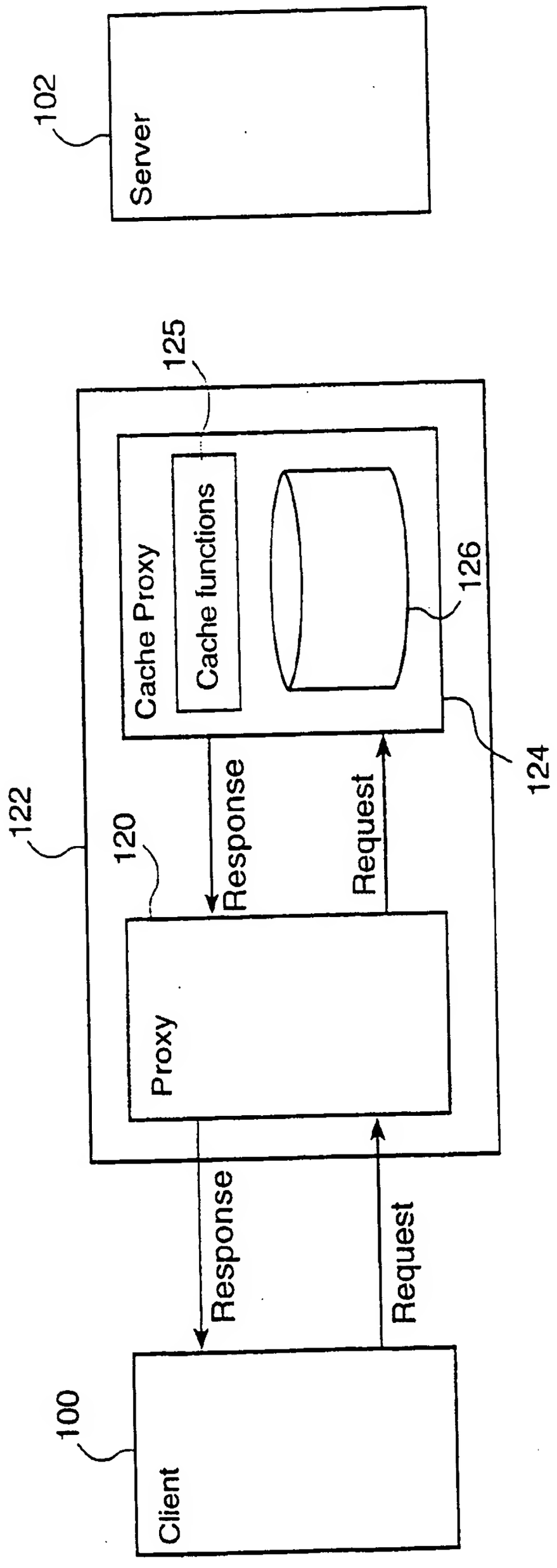

102
Server
125
Cache Proxy
Cache functions
126
124
122
120
Response
Request
Proxy
Response
Request
100
Client

FIG. 6

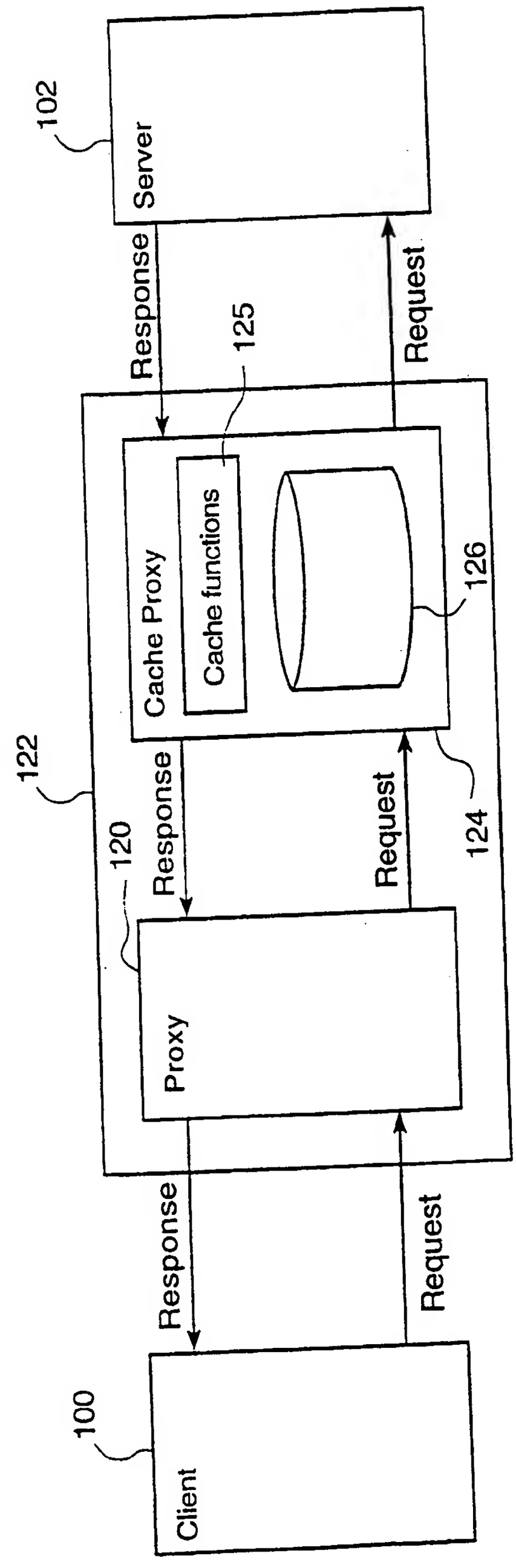

102
Server
Response
125
Request
Cache Proxy
Cache functions
126
122
120
Response
Request
124
Proxy
Response
Request
100
Client

FIG. 7

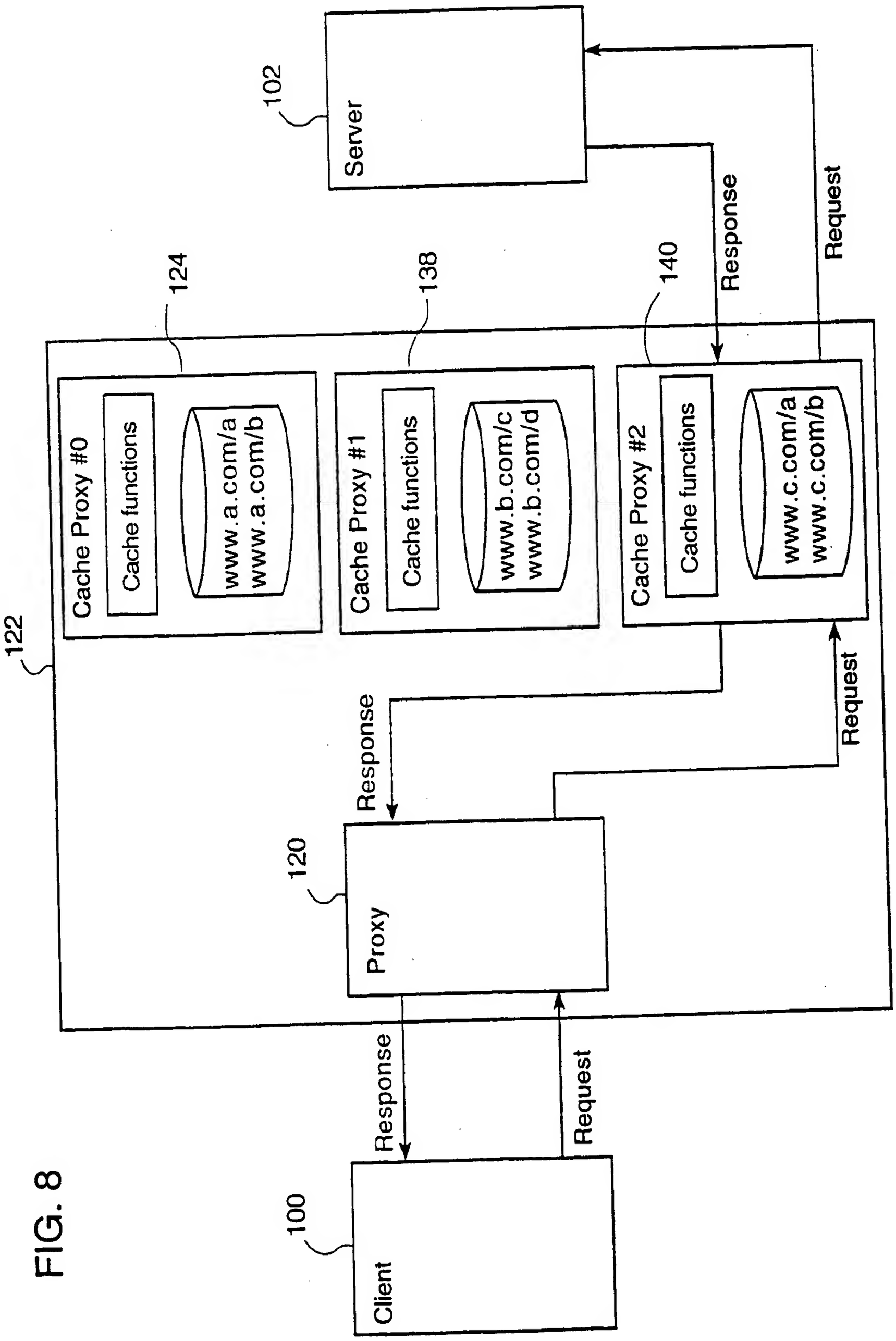

Server
102
Request
Response
140
Cache Proxy #2
Cache functions
www.c.com/a
www.c.com/b
138
Cache Proxy #1
Cache functions
www.b.com/c
www.b.com/d
124
Cache Proxy #0
Cache functions
www.a.com/a
www.a.com/b
122
Request
Response
120
Proxy
Request
Response
100
Client

FIG. 8

FIG. 9

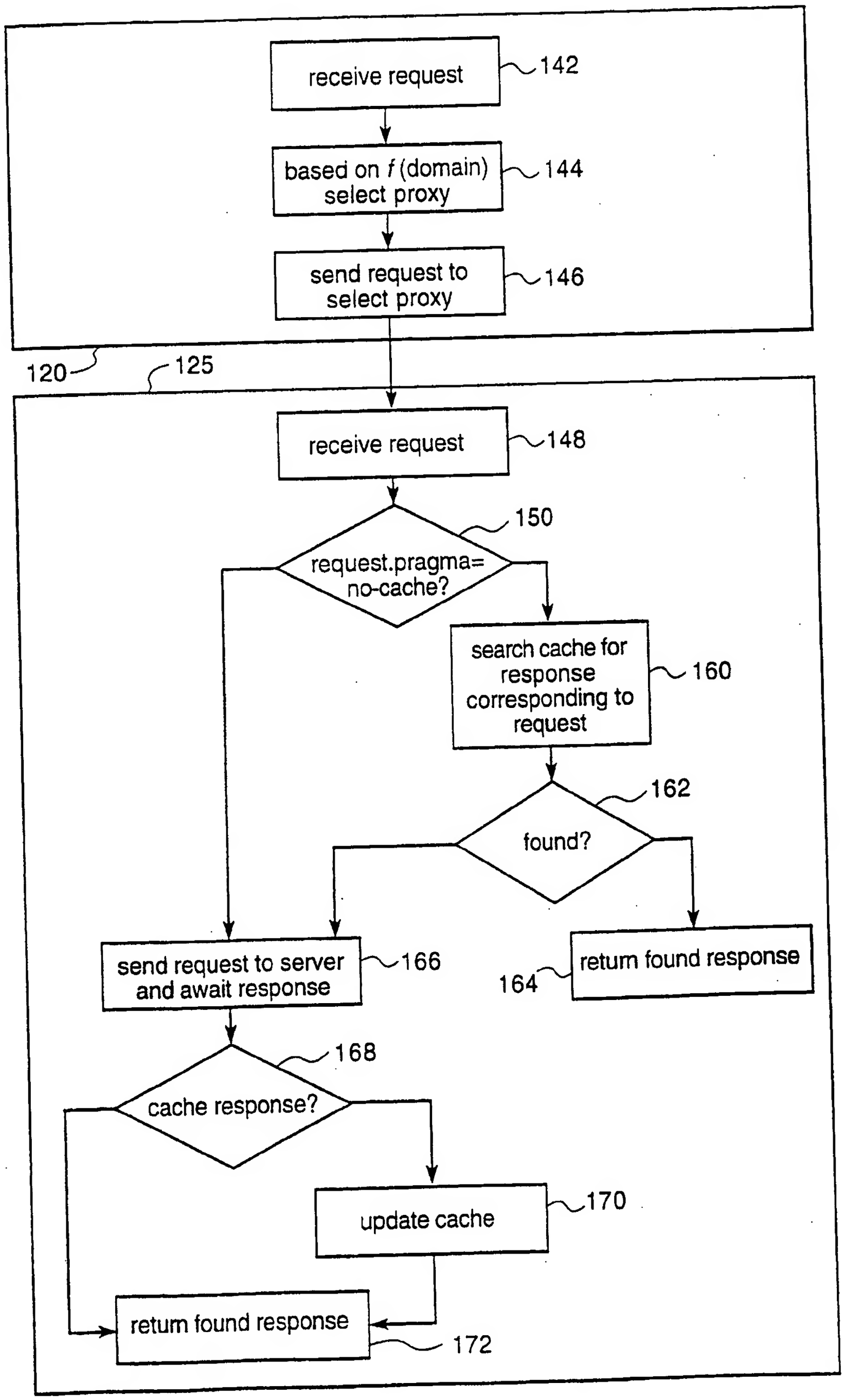

receive request
142
based on f (domain) select proxy
144
send request to select proxy
146
120
125
receive request
148
150
request.pragma= no-cache?
search cache for response corresponding to request
160
162
found?
send request to server and await response
166
return found response
164
168
cache response?
update cache
170
return found response
172

FIG. 10

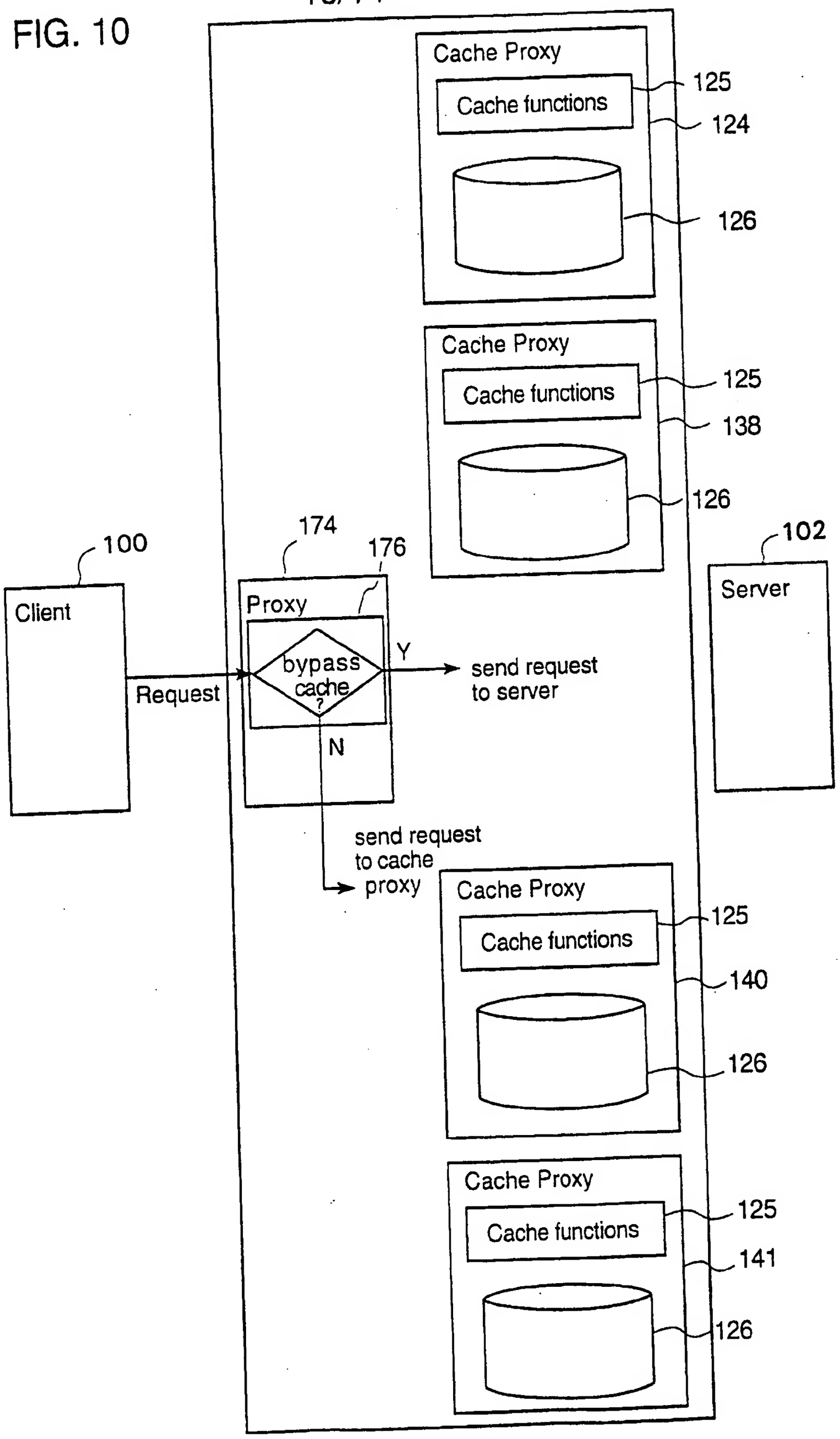

Cache Proxy
Cache functions
125
124
126
Cache Proxy
Cache functions
125
138
126
174
176
100
102
Client
Proxy
bypass cache ?
Y
send request to server
Request
N
send request to cache proxy
Server
Cache Proxy
Cache functions
125
140
126
Cache Proxy
Cache functions
125
141
126

FIG. 11

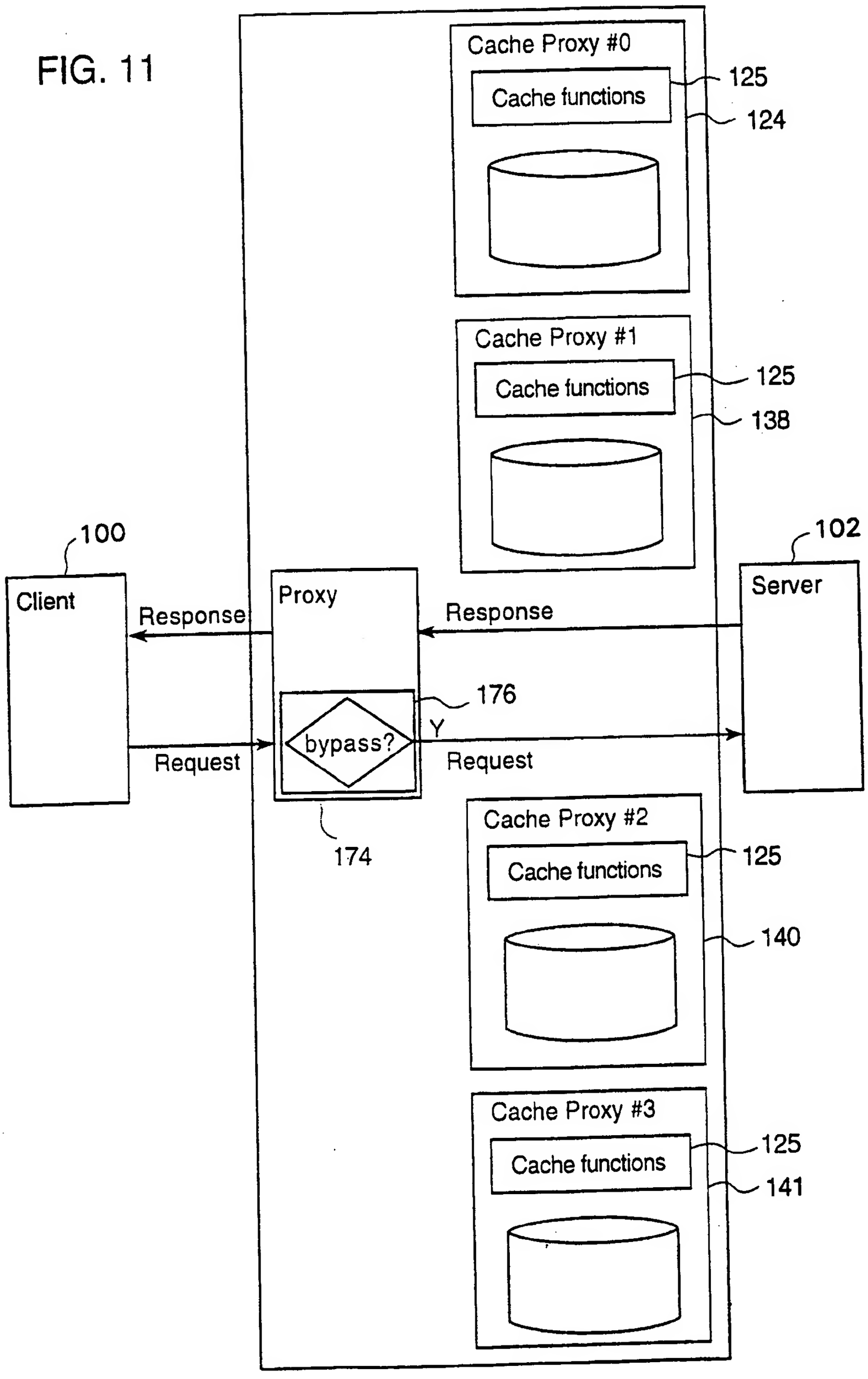

Cache Proxy #0
Cache functions
125
124
Cache Proxy #1
Cache functions
125
138
100
102
Client
Response
Proxy
Response
Server
176
bypass?
Y
Request
Request
174
Cache Proxy #2
Cache functions
125
140
Cache Proxy #3
Cache functions
125
141

FIG. 12

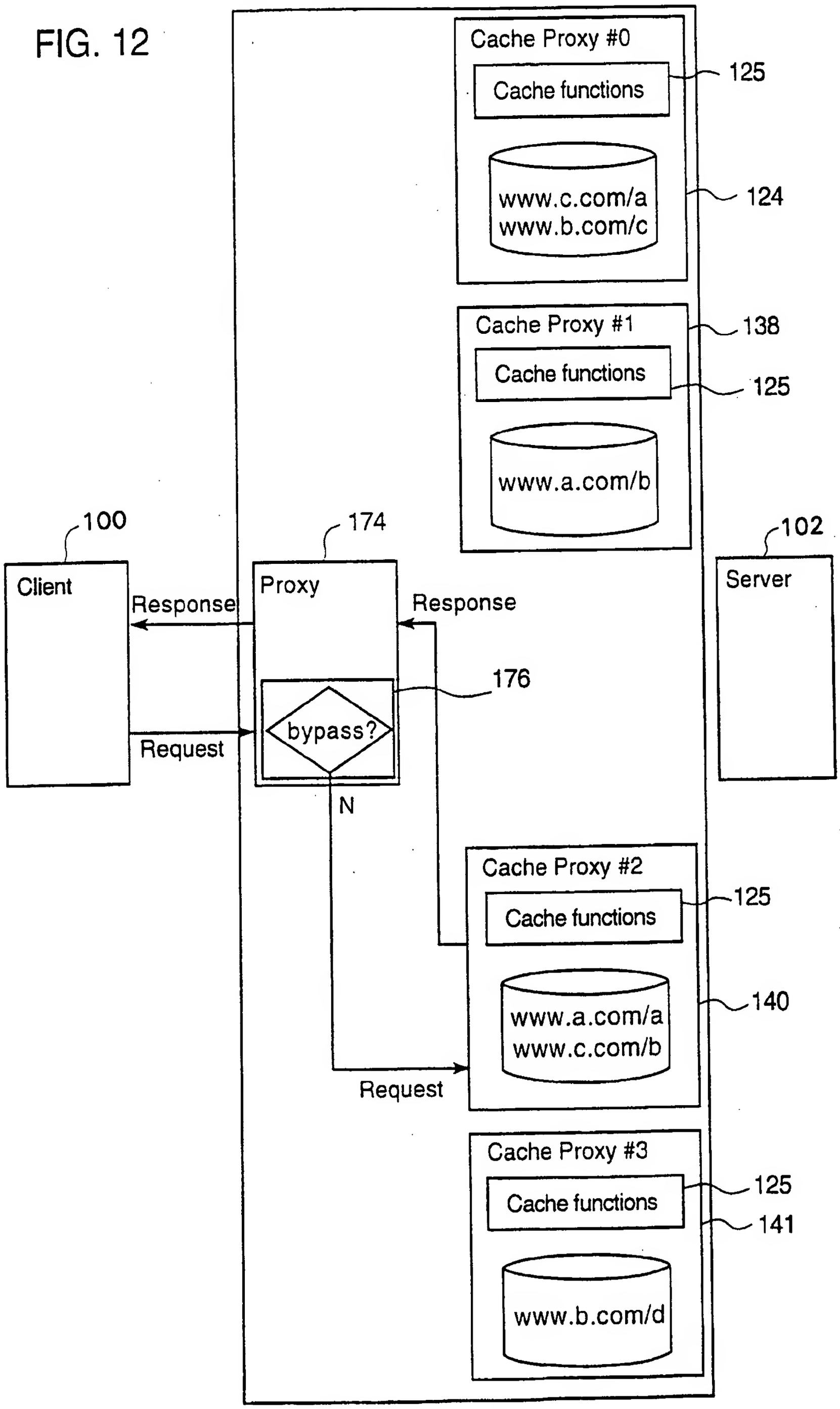

Cache Proxy #0
Cache functions
125
www.c.com/a
www.b.com/c
124
Cache Proxy #1
138
Cache functions
125
www.a.com/b
100
174
102
Client
Proxy
Server
Response
Response
176
bypass?
Request
N
Cache Proxy #2
125
Cache functions
www.a.com/a
www.c.com/b
140
Request
Cache Proxy #3
125
Cache functions
141
www.b.com/d

FIG. 13

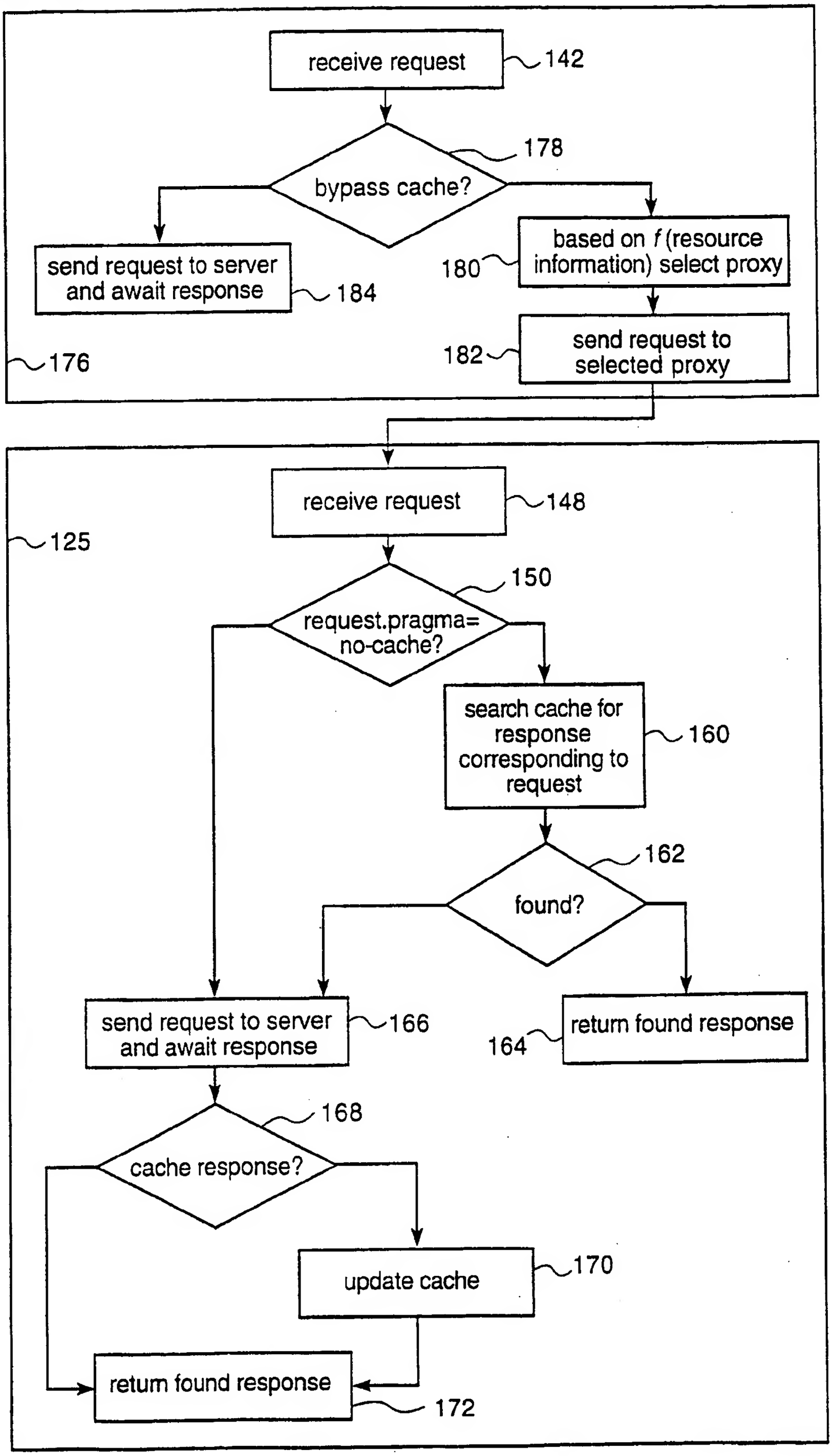
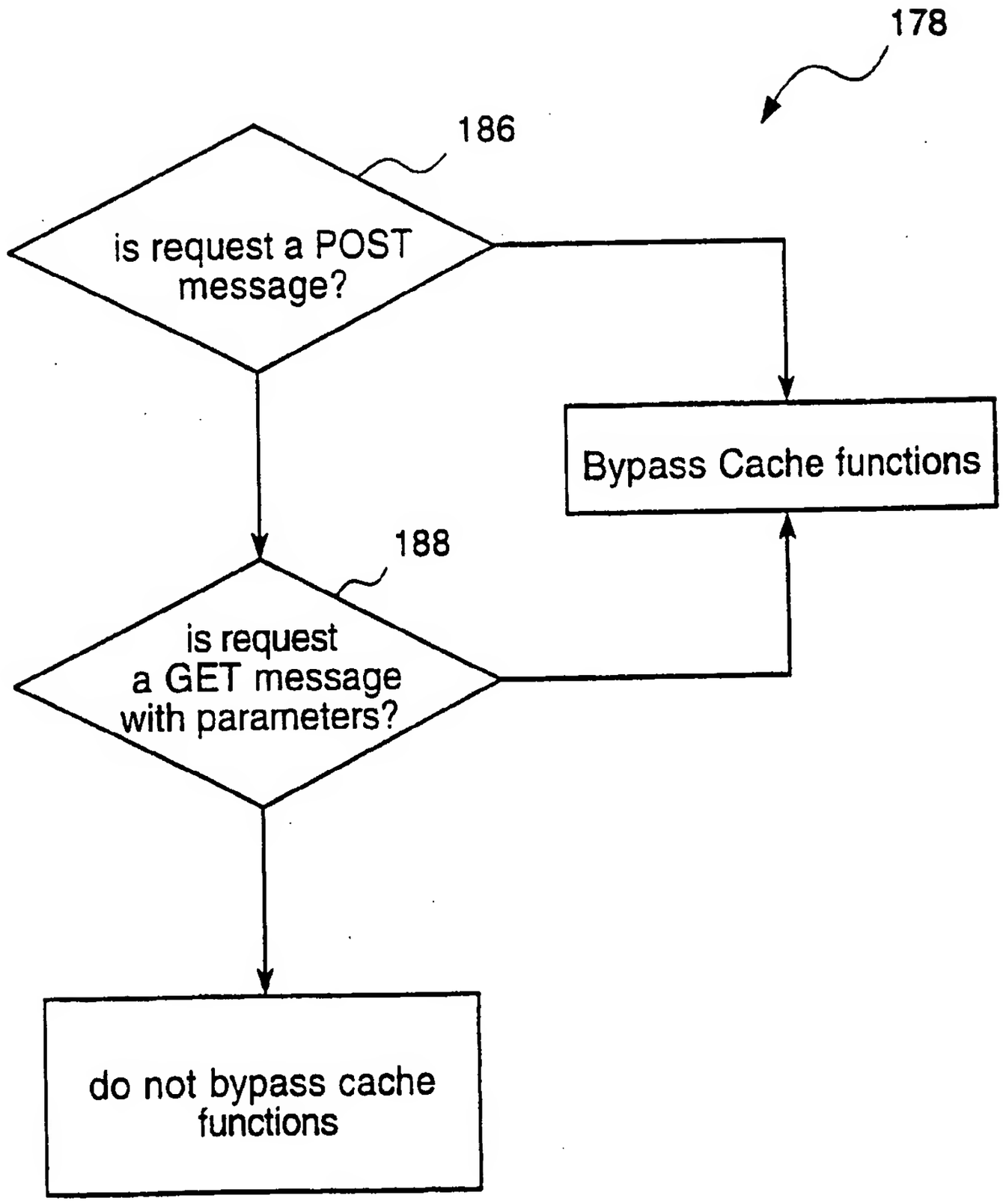

receive request
142
178
bypass cache?
send request to server and await response
184
based on f (resource information) select proxy
180
send request to selected proxy
182
176
receive request
148
125
150
request.pragma= no-cache?
search cache for response corresponding to request
160
162
found?
send request to server and await response
166
return found response
164
168
cache response?
update cache
170
return found response
172

FIG. 14


178
186
is request a POST message?
Bypass Cache functions
188
is request a GET message with parameters?
do not bypass cache functions